AI Planning
Exercise Sheet 2

Date: 06.11.2014
Students: Axel Perschmann, Tarek Saier

Exercise 2.1

(a) Transform the operator

original
$\langle \neg e \lor f, (a \rhd (b \rhd c)) \land (\neg d \rhd c) \land (\neg(\neg c \land \neg a) \rhd (d \land \neg d)) \land (d \rhd \neg e)\rangle$
(7)
$\langle \neg e \lor f, ((a \land b) \rhd c) \land (\neg d \rhd c) \land (\neg(\neg c \land \neg a) \rhd (d \land \neg d)) \land (d \rhd \neg e)\rangle$
(9)
$\langle \neg e \lor f, (((a \land b) \lor \neg d) \rhd c) \land (\neg(\neg c \land \neg a) \rhd (d \land \neg d)) \land (d \rhd \neg e)\rangle$
(8)
$\langle \neg e \lor f, (((a \land b) \lor \neg d) \rhd c) \land (\neg(\neg c \land \neg a) \rhd d) \land (\neg(\neg c \land \neg a) \rhd \neg e) \land (d \rhd \neg e)\rangle$
(9)
$\langle \neg e \lor f, (((a \land b) \lor \neg d) \rhd c) \land (\neg(\neg c \land \neg a) \rhd d) \land ((\neg(\neg c \land \neg a) \lor d) \rhd \neg e)\rangle$
2x deMorgan
$\langle \neg e \lor f, (((a \land b) \lor \neg d) \rhd c) \land ((c \lor a) \rhd d) \land ((c \lor a \lor d) \rhd \neg e)\rangle$

(b) Transform the ENF operator

original
$\langle \neg e \lor f, (((a \land b) \lor \neg d) \rhd c) \land ((c \lor a) \rhd d) \land ((c \lor a \lor d) \rhd \neg e)\rangle$
identify negative literals in coditions
$\langle \neg e \lor f, (((a \land b) \lor \neg d) \rhd c) \land ((c \lor a) \rhd d) \land ((c \lor a \lor d) \rhd \neg e)\rangle$
introduce $\hat{d}$=$\neg d$ and $\hat{e}$=$\neg e$ and add it for effects
$\langle \neg e \lor f, (((a \land b) \lor \neg d) \rhd c) \land ((c \lor a) \rhd d \land \neg \hat{d}) \land ((c \lor a \lor d) \rhd \neg e \land \hat{e})\rangle$
replace negative d and e with $\hat{d}$ and $\hat{e}$ in conditions
$\langle \hat{e} \lor f, (((a \land b) \lor \hat{d}) \rhd c) \land ((c \lor a) \rhd d \land \neg \hat{d}) \land ((c \lor a \lor d) \rhd \neg e \land \hat{e})\rangle$
(8)
$\langle \hat{e} \lor f, (((a \land b) \lor \hat{d}) \rhd c) \land ((c \lor a) \rhd d) \land ((c \lor a) \rhd \neg \hat{d}) \land ((c \lor a \lor d) \rhd \neg e) \land ((c \lor a \lor d) \rhd \hat{e})\rangle$

Exercise 2.2

(a-c)

Listing 1: set cover problem as a PDDL domain

```
(define (domain set-cover)
    (:requirements :adl)
    (:types set elem)
    (:predicates (contains ?s - set ?e - elem)
        (selected ?s - set)
        (covered ?e - elem))
    (:action select-set
        :parameters(?s - set)
        :precondition (not (selected ?s))
        :effect (and (selected ?s)
                     (forall (?e - elem)
                        (when (contains ?s ?e)
                            (covered ?e))))))
```

Listing 2: set cover instance as a PDDL problem

```
(define (problem set-cover-i-1)
    (:domain set-cover)
    (:objects e1 e2 e3 e4 - elem s1 s2 s3 s4 s5 - set)
    (:init (contains s1 e1)
        (contains s2 e2)
        (contains s2 e3)
        (contains s3 e4)
        (contains s4 e1)
        (contains s4 e2)
        (contains s5 e3)
        (contains s5 e4))
    (:goal (and (covered e1)
                (covered e2)
                (covered e3)
                (covered e4))))
```

Listing 3: solving set cover instance with fast-downward

```
$ ./fast-downward.py scprob.pddl --search "astar(blind())"
[...]
select-set s4 (1)
select-set s5 (1)
Plan length: 2 step(s).
Plan cost: 2
Initial state h value: 1.
Expanded 6 state(s).
Reopened 0 state(s).
Evaluated 16 state(s).
Evaluations: 16
Generated 21 state(s).
Dead ends: 0 state(s).
Expanded until last jump: 1 state(s).
Reopened until last jump: 0 state(s).
Evaluated until last jump: 6 state(s).
Generated until last jump: 5 state(s).
Number of registered states: 16
Search time: 0s
Total time: 0s
```

```
Solution found.
Peak memory: 2924 KB
$ cat sas_plan
(select-set s4)
(select-set s5)
; cost = 2 (unit cost)
```

(d)

- Optimal planning: every possible plan is considered. It is guaranteed that the optimal plan (lowest cost) can be identified.
- Satisficing planning: it is sufficient to identify a plan that leads to the goal.
- Arbitrary set covers: as long as every element is covered we're satisfied.
- Cardinality minimal set covers: the goal is to pick $\mathcal{C} \subseteq \mathcal{S}$ such that every element is covered and $|\mathcal{C}|$ is minimal.

Since our formalization does only define complete coverage of elements as the goal, cardinal minimality can only be achieved by applying optimal planning. Since fast-downward increments the cost of a plan for each applied operation and the only operator defined is selecting an element of $\mathcal{S}$, cost of a plan equals to $|\mathcal{C}|$. Finding the optimal plan therefore means finding the minimal $\mathcal{C} \subseteq \mathcal{S}$ such that every element is covered.

In our formalization plan existence is equivalent to the existence of a set cover. If we were to include an element $e_5 \in \mathcal{U}$ such that $\forall S_i \in \mathcal{S} : e_5 \notin S_i$ there would be no possible set cover and therefore no existing plan that solves the problem.